Big Data
algoritmusok és szoftver-rendszerek

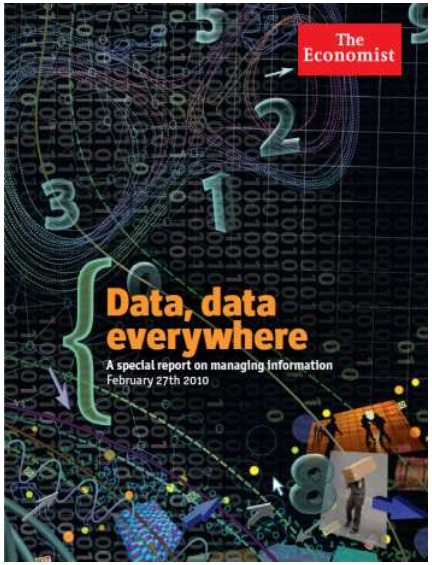
Benczúr András

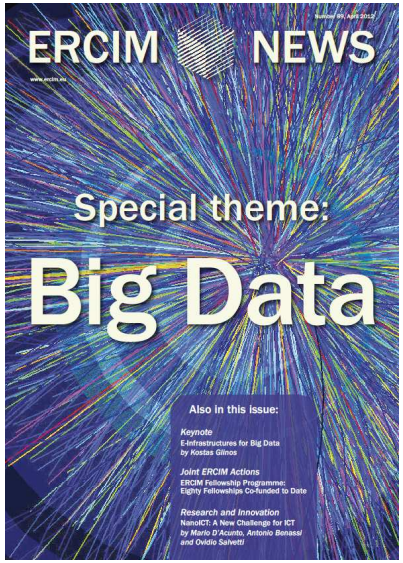
MTA SZTAKI

benczur@sztaki.mta.hu

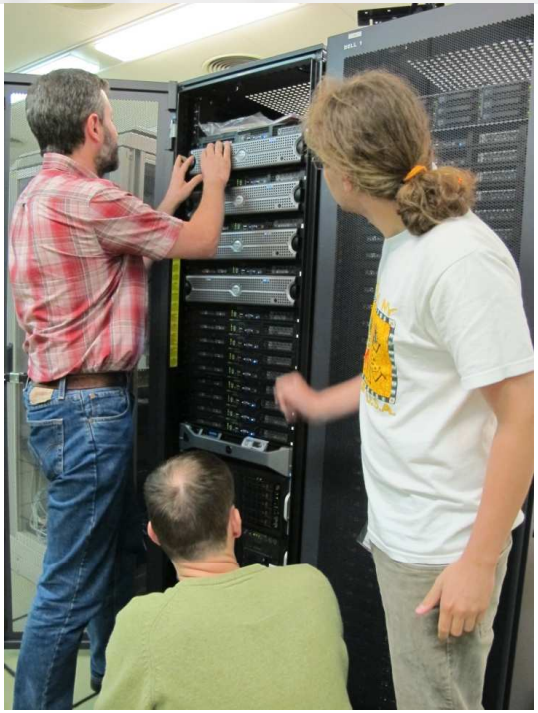
http://datamining.sztaki.hu

Big Data


The Economist
Data, data everywhere
A special report on managing information
February 27th 2010


ERCIM NEWS
Special theme:
Big Data
Also in this issue:
Keynote
E-Infrastructures for Big Data
by Kostas Glinos
Joint ERCIM Actions
ERCIM Fellowship Programme:
Eighty Fellowships Co-funded to Date
Research and Innovation
NanoICT: A New Challenge for ICT
by Mario D'Acunto, Antonio Bonassi and Ovidio Salvetti


nature
BIG DATA
SCIENCE IN THE PETABYTE ERA

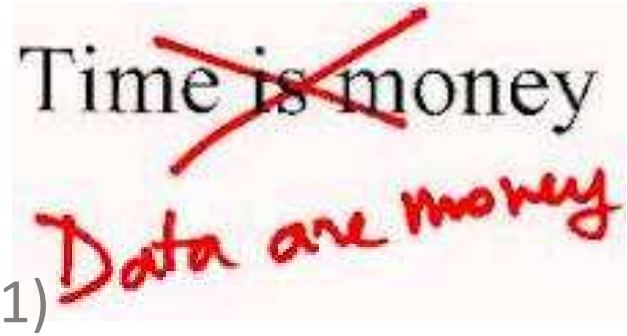



The New York Times
August 12, 2012
BIG DATA
Lloyd Miller for The New York Times


hadoop
S3
memcached
Sybase IQ
Greenplum
HYPERTABLE
InfiniDB
TERADATA
VERTICA
Sun
ORACLE
membase
HIVE
HBASE
INFOBRIGHT
twinFin
Microsoft SQL Server
Redis
CouchDB
DB2
Cassandra
mongoDB

Big Data – the new hype

- “big data” is when the size of the data itself becomes part of the problem
- “big data” is data that becomes large enough that it cannot be processed using conventional methods



- Google sorts 1PB in 33 minutes (07-09-2011)
- Amazon S3 store contains 499B objects (19-07-2011)
- New Relic: 20B+ application metrics/day (18-07-2011)
- Walmart monitors 100M entities in real time (12-09-2011)

Source: The Emerging Big Data slide from the Intelligent Information Management DG INFSO/E2 Objective ICT-2011.4.4 Info day in Luxembourg on 26 September 2011

Adatbányászat, Big Data

- Adatbányászat: Hasznos (meglepő?) tudás kinyerése nagy adattömegből
- Technikák

- Algoritmusok (nagy méret)
- Adatbázisok (elrendezés, hozzáférés)

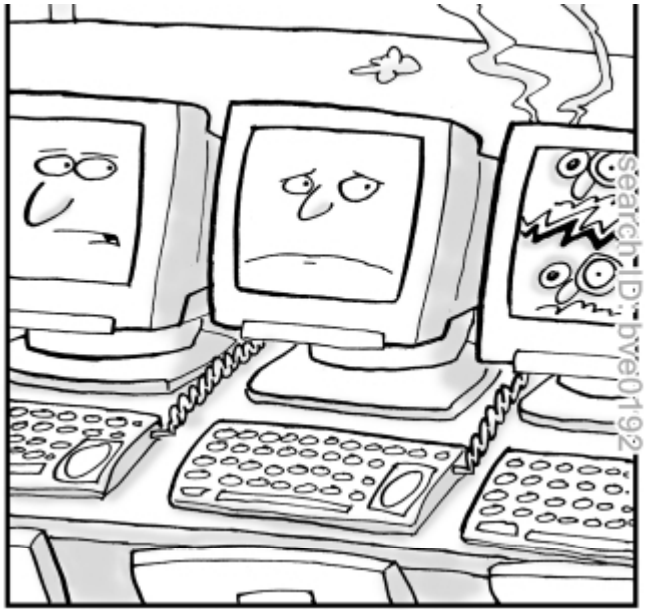
- Mesterséges Intelligencia és Gépi Tanulás (modellek)
- Statisztika (hipotézisvizsgálat)

- Big Data – minden még nagyobb

- Algoritmusok (elosztott, MapReduce, Cloud)
- Adatbázisok (elosztott, NoSQL)
- Okostelefonok, közösségi média (Facebook, Twitter, …)

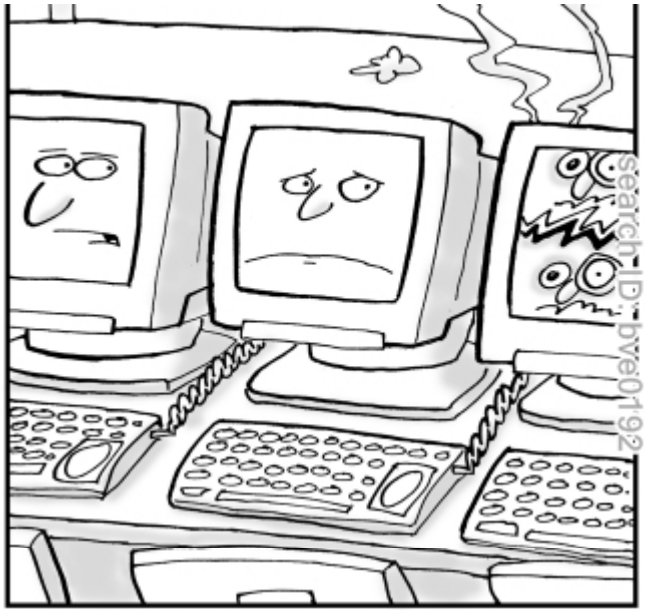
- Mesterséges Intelligencia és Gépi Tanulás – ajánló rendszerek, hálózatok
- Statisztika

Elosztott rendszerek Murphy törvénye



Elosztott rendszerek Murphy törvénye

Fox&Brewer “CAP Tétel”:
C-A-P: kettőt választhatunk!

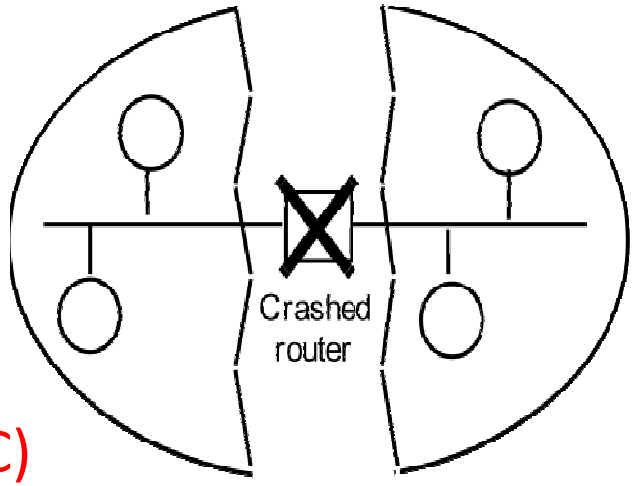


Végül konzisztenssé válhat – eventual consistency

Mi történik, ha szétesik a rendszer?

CAP tétel bizonyítás

- Partition (P): a jobb oldalra beírt új értéket nem ismeri a bal oldal
- Ha azonnal kérdezünk a bal oldalon (availability), akkor hibás a válasz
- Vagy availability (A), vagy konzisztencia (C)



- Végül lehet konzisztens (eventual consistency)
- A kapcsolat helyreállása után lehet adatot cserélni

Duplikátum-keresés: erősebb korlátok!

name	e-mail	ID
Mary Smith	m.smith@mail-1.com	50071
Mary Doe	mary@mail-2.com	50071
M. Doe	mary@mail-2.com	79216
M. Smith	m.smith@mail-1.com	34302

Sidló, B, Garzó, Molnár, **Infrastructures and bounds for distributed entity resolution.** *QDB 2011*

Duplikátum-keresés: erősebb korlátok!

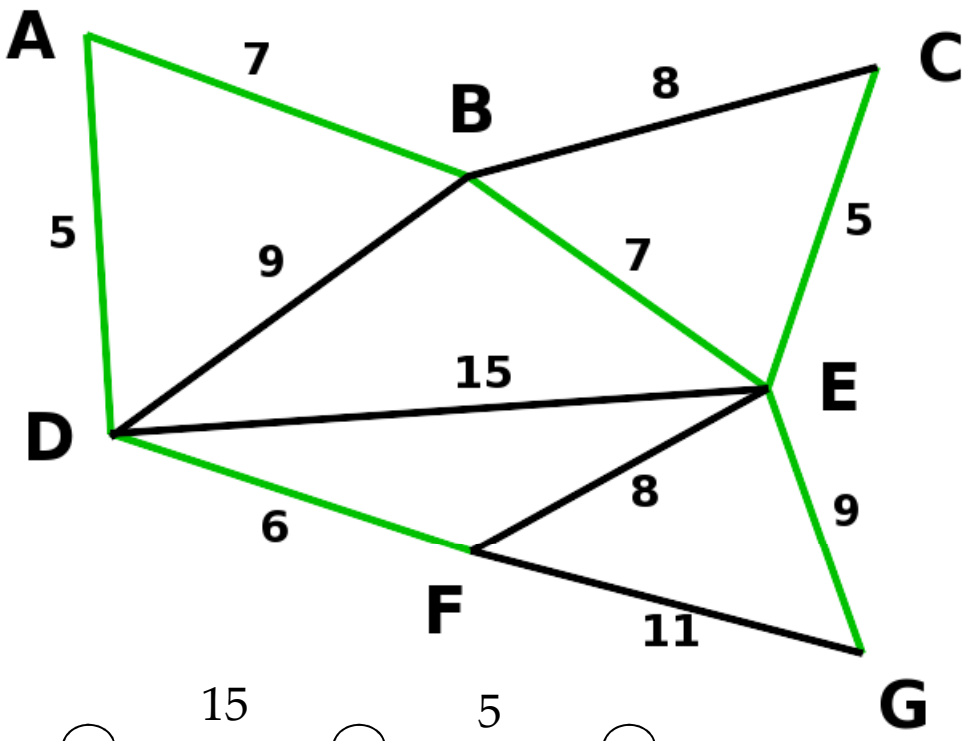
- Halmaz metszet kommunikációs bonyolultsága $\Theta(n)$ bit [Kalyanasundaram, Schintger 1992]
- Következmény: több szerveren elosztott adatok esetén $\Theta(n)$ kommunikáció eldönteni, hogy van-e duplikátum!
- Javasolt módszerek: „Blocking”

  [Whang, Menestrina, Koutrika, Theobald, Garcia-Molina. ER with Iterative Blocking, 2009, stb.]
- Legjobb esetben is *minden* adatot ki kell cserélni
- Kapcsolódó terület: Locality Sensitive Hashing
  - nincs „minimum”, azaz koordináta egyezés LSH
  - hasonló a Donoho Zero „norm” (nem-0 koordináták száma) negatív eredményekhez

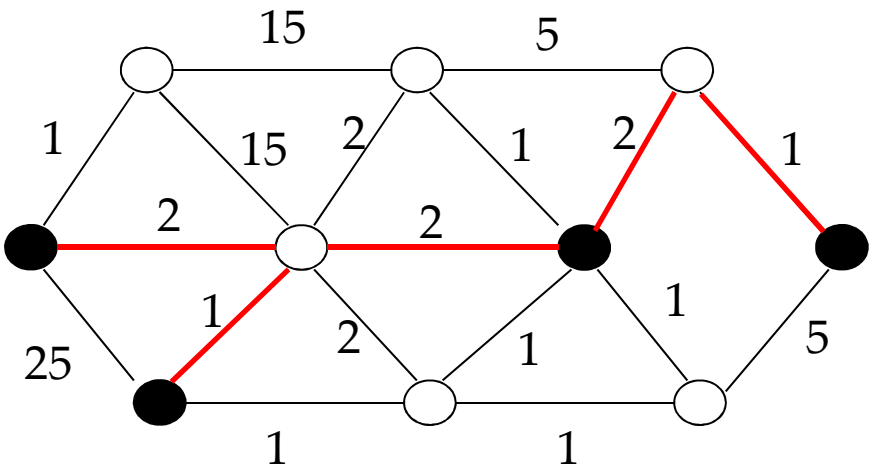
Sidló, B, Garzó, Molnár, **Infrastructures and bounds for distributed entity resolution.** *QDB 2011*

Gráfalgoritmusok őstörténete: P, NP

- P: Gráfbejárás;
  Feszítőfa



- NP: Steiner fa

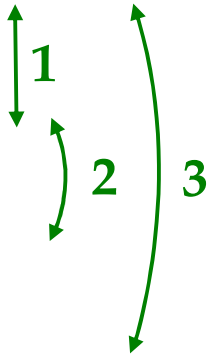


Kit érdekel ez még ma?

Kép-szegmentálás

Azonosság-feloldás

name	e-mail	ID
Mary Smith	m.smith@mail-1.com	50071
Mary Doe	mary@mail-2.com	50071
M. Doe	mary@mail-2.com	79216
M. Smith	m.smith@mail-1.com	34302

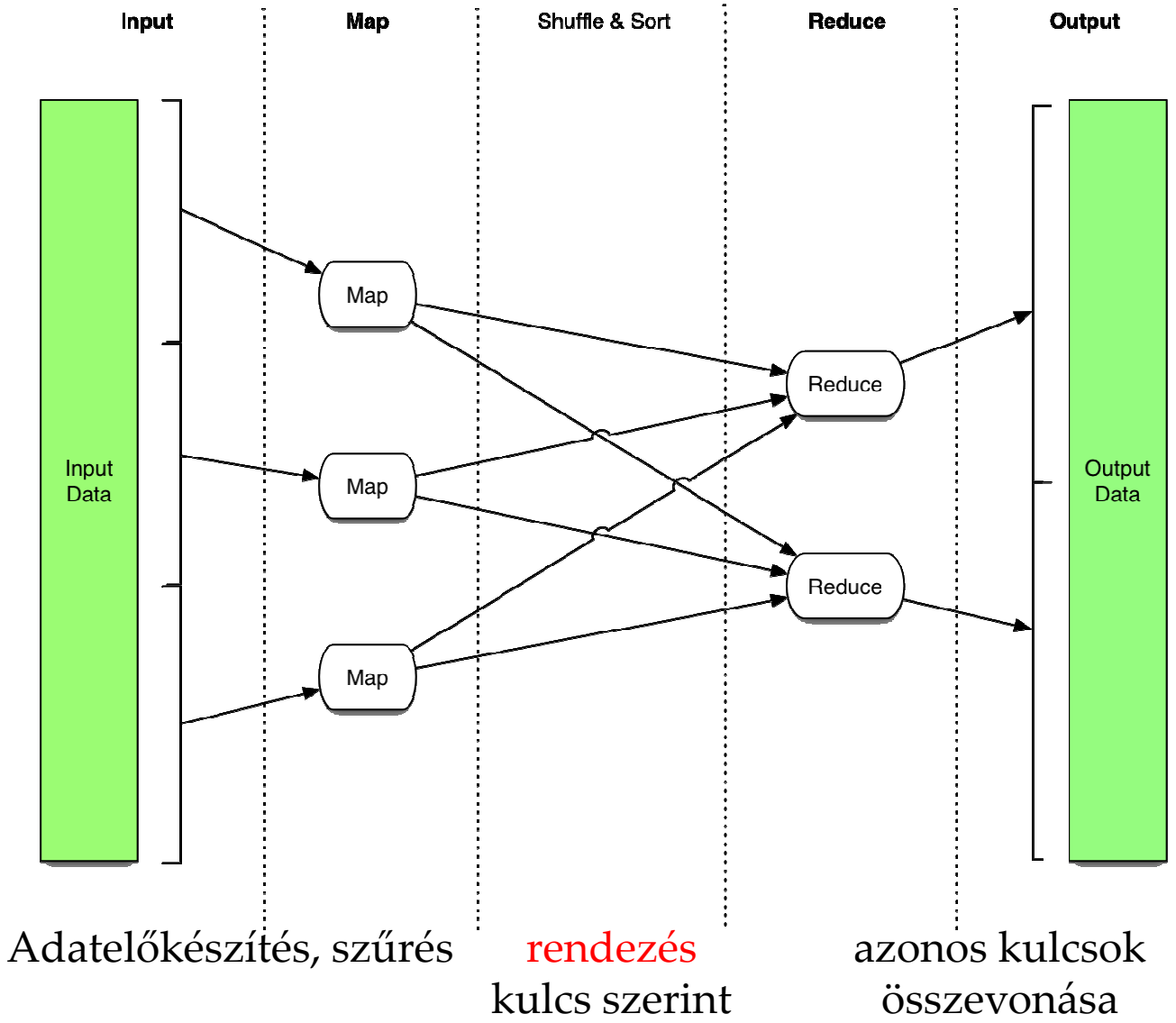


MapReduce

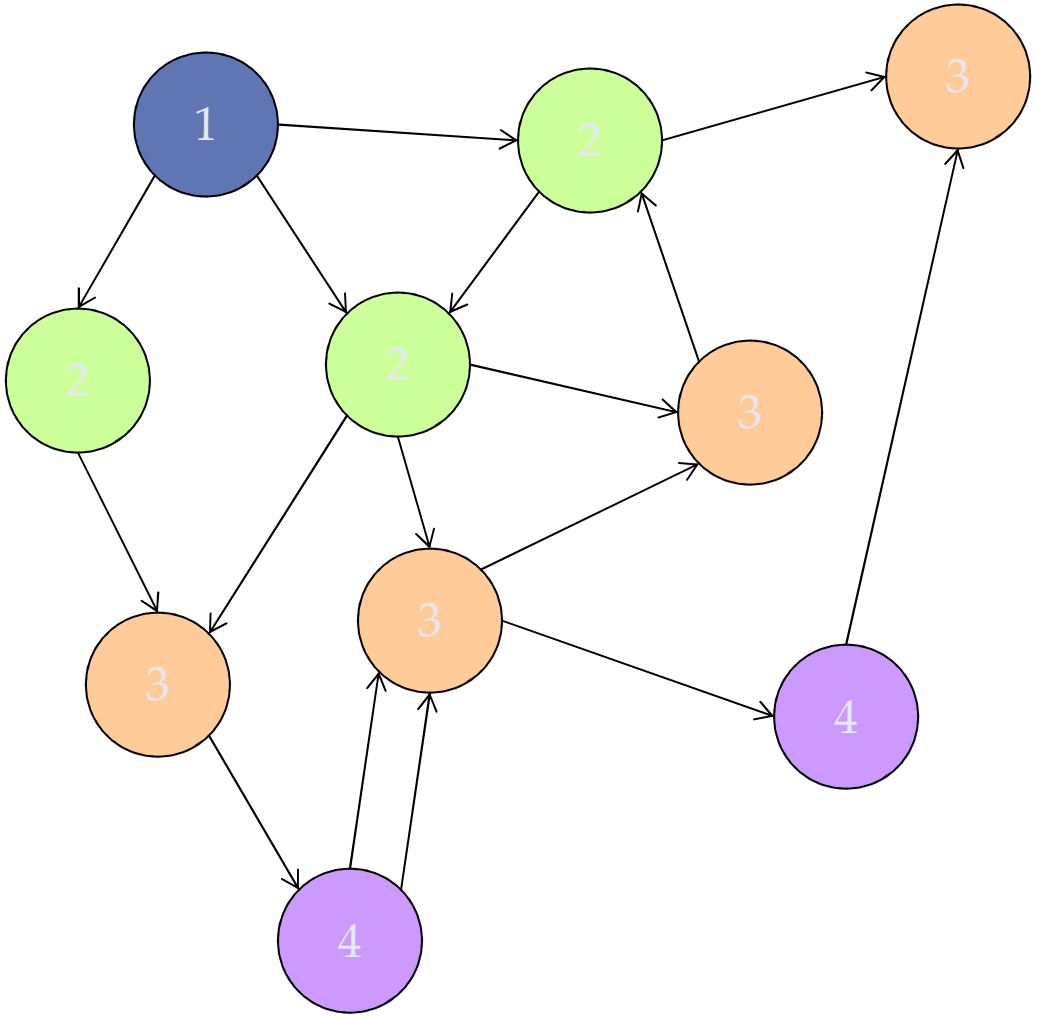
- Google technológia

  MapReduce: simplified data processing on large clusters. J Dean, S Ghemawat - Communications of the ACM, 2008 [OSDI 2004]
- Hadoop: Yahoo! által indított open source
- HDFS: Hadoop Distributed File System
- MapReduce: kétfázisú algoritmus környezet

Map/Reduce programozási modell



Szélességi bejárás



Szélességi bejárás

- MAP:
  - Minden *n* csúcs távolsága starttól (D); ki-éllista
- $\forall p \in$ ki-él(n): emit (*p*, D+1)
- Reduce
  - p szerint rendezve kapja
  - kiválasztja a legkisebb értéket (új távolság)
  - mindent kiír diszkre, újraindul
- Végetér, ha egy iterációban nincs változás
- Össze kell rendelni a ki-él(n)-t és az új D-t
  - Megoldás: emit (n, éllista(n)) is kell!
- Élsúlyokkal?
  - A fenti a Bellman-Ford algoritmus
  - Dijkstra hatékonyabb, mert csak a „határon” számol

MapReduce BFS kód

```
public static void main(String[] args) {
    String[] value = {
    // key | distance | points-to
          "1|0|2;4",
          "2|"+Integer.MAX_VALUE+"|1;3;4",
          "3|"+Integer.MAX_VALUE+"|2",
          "4|"+Integer.MAX_VALUE+"|1;3",
    };

    mapper(value);
    reducer(collect.entrySet());
}
```

	1	2	3	4
1	0	1	0	1
2	1	0	1	1
3	0	1	0	0
4	1	0	1	0

MapReduce BFS kód

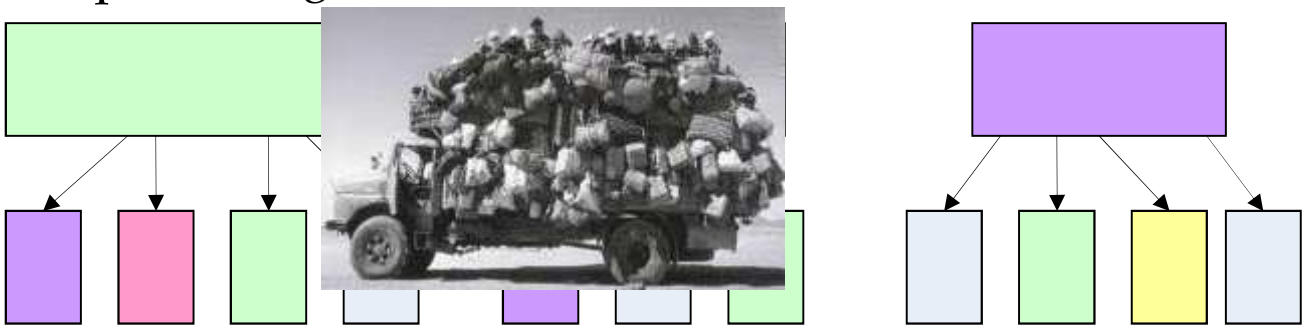
```
private static void
    reducer(Set<Entry<String, ArrayList<String>>> entrySet) {
        for (Map.Entry<String, ArrayList<String>> e : entrySet) {
            Iterator<String> values = e.getValue().iterator();
            int minDist = Integer.MAX_VALUE;
            String link_list = "";
            while (values.hasNext()) {
                String[] dist_links =
                    values.next().toString().split("[|]");
                if (dist_links.length > 1)
                    link_list = dist_links[1];
                int dist = Integer.parseInt(dist_links[0]);
                minDist = Math.min(minDist, dist);
            }
        System.out.println(e.getKey() + " - D " + (minDist + " | " + link_list));
        }
    }
}
```

MapReduce BFS kód

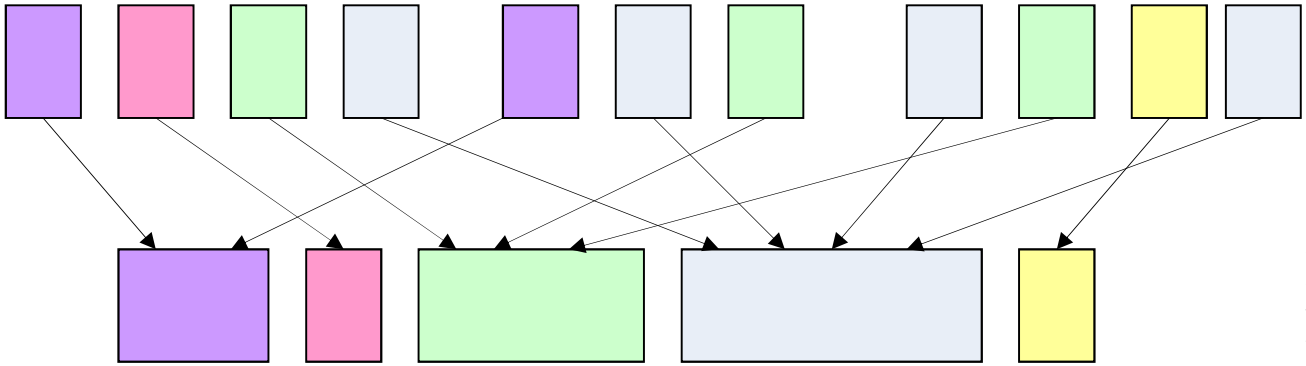
```
private static void mapper(String[] value) {
	for (int i = 0; i < value.length; i++) {
		String line = value[i].toString();
		String[] keyVal = line.split("[|]");

		String Key = keyVal[0];
		String sDist = keyVal[1];
		String[] links = null;
		if (keyVal.length > 2) {
			links = keyVal[2].split(";");
			int Dist = Integer.parseInt(sDist);

			if (Dist != Integer.MAX_VALUE)
				Dist++;
			for (int x = 0; x < links.length; x++) {
			if (links[x] != "") {
				ArrayList<String> list;
				if (collect.containsKey(links[x])) {
				list = collect.get(links[x]);
				} else {
				list = new ArrayList<String>();
				}
			list.add(Dist + "|");
			collect.put(links[x], list);
		}
	}

	ArrayList<String> list;
	if (collect.containsKey(Key)) {
		list = collect.get(Key);
		} else {
		list = new ArrayList<String>();
		}
		list.add(sDist + "|" + keyVal[2]);
		collect.put(Key, list);
		}
	}
}
```

MapReduce BFS

Map: távolság + 1 átadása a szomszédoknak



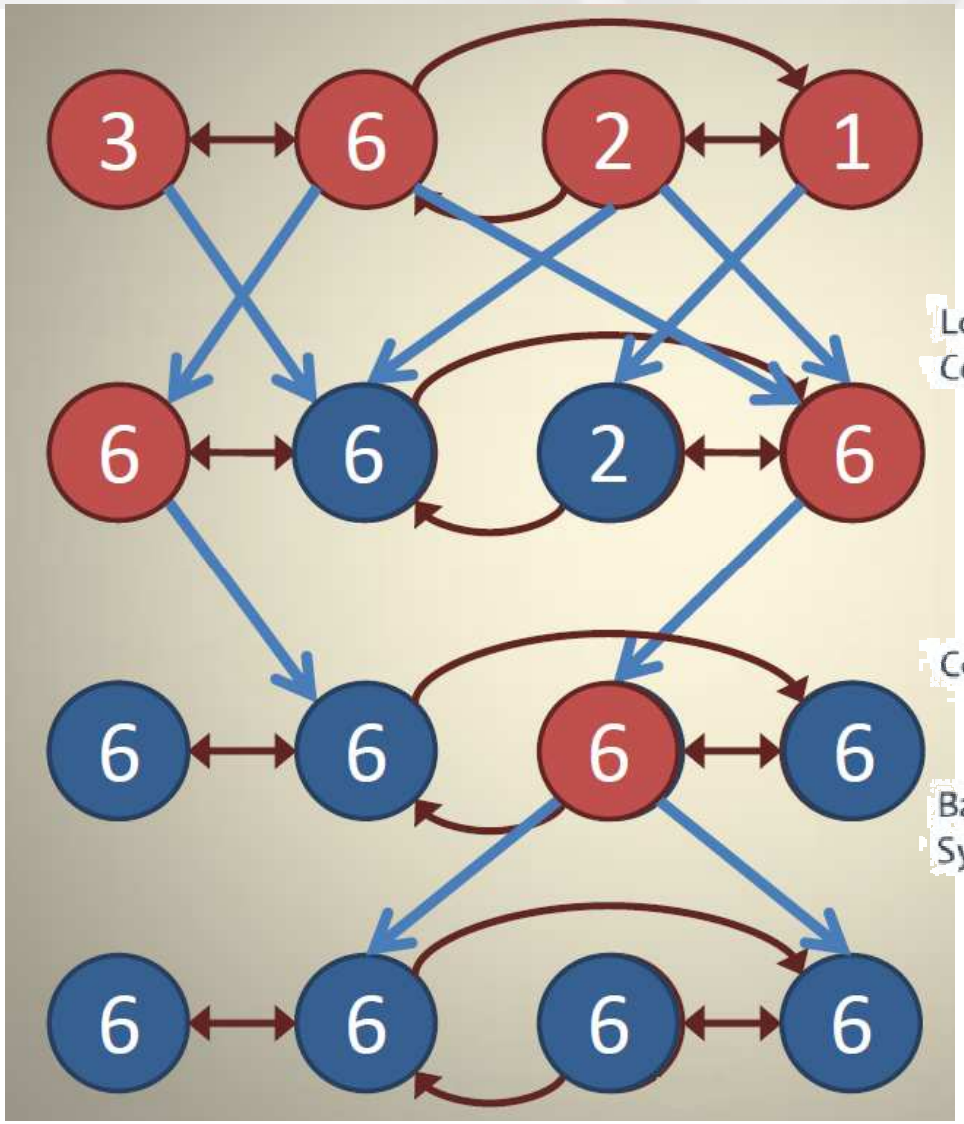
Reduce: minimum számítása

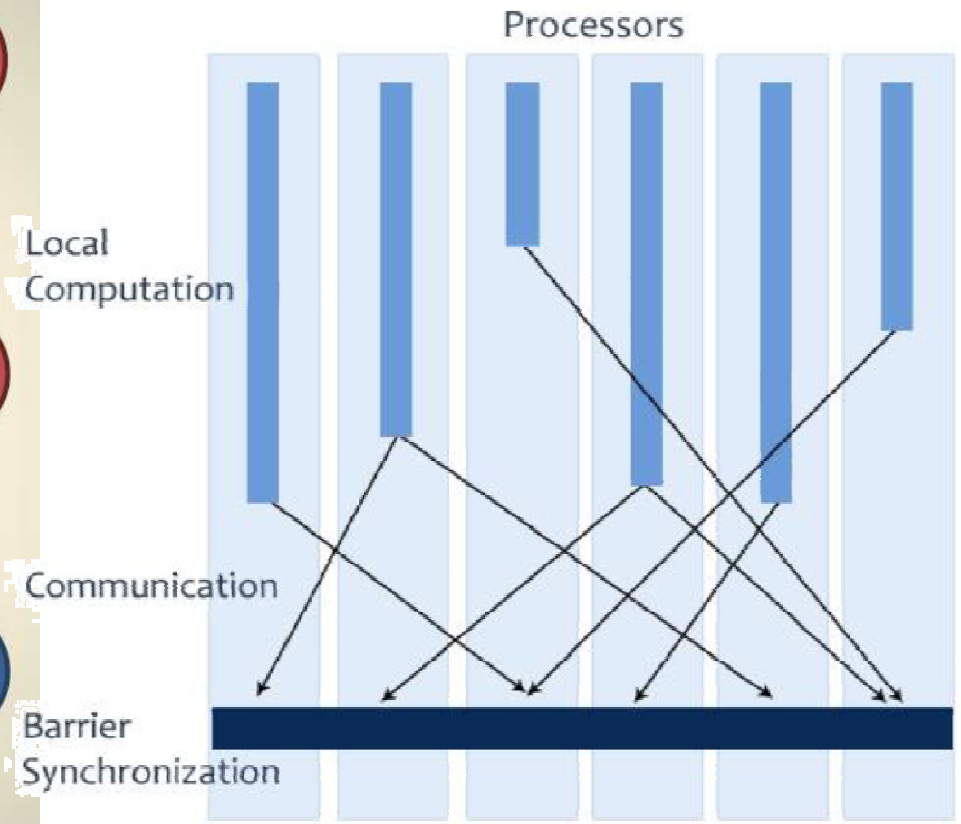


Iteráció, amíg konvergál

...

Bulk Synchronous Parallel (BSP) komponensek

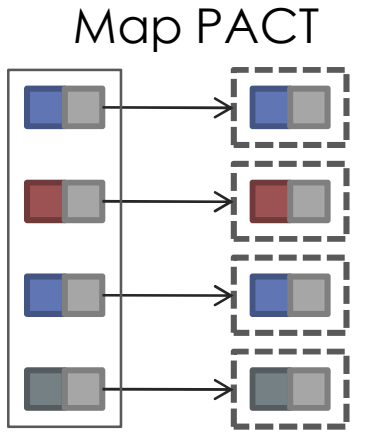


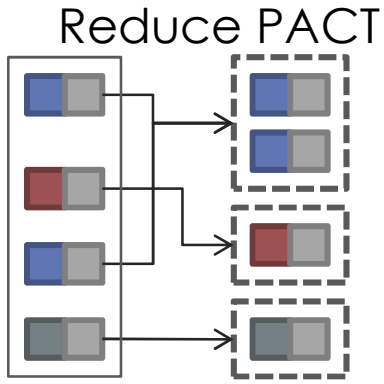


Google Pregel (nem publikus)
GraphLab (C++, több mint BSP)
Giraph, HAMA, …

Parallelization Contract, BSP és a Join művelet

- Map PACT (PArallelization ContracT)
  - Minden rekord egy csoport
  - Minden csoportot külön dolgozhatjuk fel
- Reduce PACT
  - Egyik attribútum a kulcs
  - Azonos kulcs egy csoporthoz tartozik



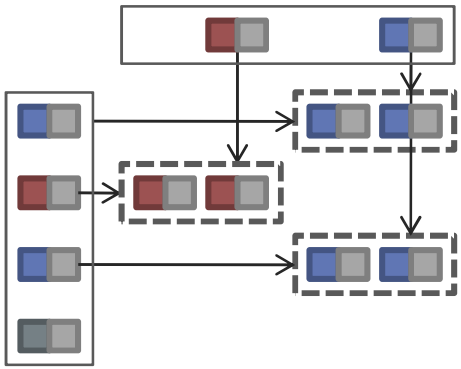


Parallelization Contract, BSP és a Join művelet

Join PACT

Minden azonos kulccsal rendelkező pár egy csoport

(equi-join)



BSP

Csúcsok és Élek

Kulcs a csúcs ID

Egy csúcs szomszédjainak összegyűjtése

A Stratosphere rendszer

- PACT programozási modell
- Végrehajtás optimalizáció, mint hagyományos adatbázis-kezelőknél
- Alacsony szintű adatfolyam engine (Nephele)
- Képes adatcsatornát (memória, diszk, hálózat) választani, adatot memóriában tartani, pl. MapReduce-t hatékonyan iterálni
- Elméletben …

Algoritmusok adatfolyamokon

- Számítási modellek
  - Belső tár (P, NP)
  - Külső tár
  - Adatfolyam
- Algoritmus típusok
  - Determinisztikus
  - Randomizált (Las Vegas ill. Monte Carlo)
  - Közelítő
- Mértékek
  - Idő, adatok elolvasásának száma
  - Tárhely

Különböző értékek száma

- Feladat
  - Input $X = x_1, x_2, ..., x_n$
  - Értékkészlet $U = \{0, 1, 2, ..., m-1\}$
  - *D(X)* – különböző értékek száma X-ben
  - Gond: Nagy értékkészlet (szöveg, URL, IP+port stb)
- Nézzük meg különböző …
  - számítási modellekben
  - algoritmikus modellekben
- … hogy megértsük az adatfolyam modell nehézségét
- Történet: Alon, Mathias, Szegedy 1996 Self-join vagy második momentum méretére kommunikációs korlátok, véletlen közelítő algoritmusok, Gödel Prize 2005

Belső tárban: hash táblával

- Táblaméret $r = \vartheta(n)$, hash függvény $h:U \rightarrow [1..r]$
- Inicializálni $A[1..r]$ tömböt; $D = 0$
- Minden input értékre
  - Ellenőrizni, hogy i szerepel-e az $A[h(i)]$-ban tárolt listában
  - Ha nem, $D \leftarrow D+1$, és i hozzáadása az $A[h(i)]$-ban tárolt listához
- Output D

- „Véletlen” h: kevés ütközés, legtöbb lista hossza $O(1)$
- Így
  - Idő $O(n)$ [várható]
  - Tár $O(n)$

Külső tár algoritmus modell

- Ha az input nem fér el a belső tárban
- *M* memóriaméret
- Input méret $n >> M$
- Adat diszken
  - Diszken egy blokkban $B << M$ adat
  - Egység lépés egy blokk diszk és memória közötti mozgatása
- Memória műveletek ingyen vannak!

Miért blokkok?? Memória ingyen??

- Blokk írás/olvasás?
  - Átviteli sebesség ≈ 100 MB/sec (kb)
  - Blokk méret ≈ 100 KB (kb)
  - Blokk átvitel (kb 1 ms) << Seek idő (kb 10 ms)
  - Tehát – csak a seek-ek száma érdekes
- Lineáris olvasás
  - még jobb, mert kevesebb seek
- Memória miért van ingyen?
  - Processzor sebesség – pár GHz
  - Seek idő kb 10 ms

„Numbers Everyone Should Know”

Jeff Dean, Google

RAM

- L1 cache reference 0.5 ns
- L2 cache reference 7 ns
- Main memory reference 100 ns
- Read 1 MB sequentially from memory 250,000 ns

Intra-process communication

- Mutex lock/unlock 100 ns
- Read 1 MB sequentially from network 10,000,000 ns

Disk

- Disk seek 10,000,000 ns
- Read 1 MB sequentially from disk 30,000,000 ns

Miért nem jó külső tárban a hash tábla?

- Probléma
  - Hash tábla A nem fér a memóriába
  - Minden inputra A véletlen eleme kell
  - Minden elem véletlen diszk seek
  - Lépésszám – $\Omega(n)$ diszk blokk hozzáférés
- Lineáris idő – $O(n/B)$ lenne ebben a modellben
- MergeSort a jó megoldás

Mintavételezés: ha az adat túl nagy

- Előny – szublineáris tár
  - Több adat, mint diszk (ez azért ritka ☺)
  - Túl gyorsan jön az adat, nem tudjuk kiírni
- Ára – közelítési hiba
- Szokásos naiv megoldás
  - Véletlen minta R (mérete r) az n hosszú X-ből
  - Mintában D(R)
  - Becslés $\hat{D} = D(R) \times n / r$
- Baj – ritka értékek alulreprezentáltak!
- Van-e jobb megoldás??

Mintavételezés negatív eredmény

Tétel: ha E becsüli D(X)-t r<n érték vizsgálatával, ahol a mintavételezés függhet a látott adatoktól is, akkor E relatív hibája

$$\sqrt{\frac{n-r}{2r}\ln\frac{1}{\delta}}$$

legalább δ hibával, ahol $\delta > e^{-r}$.

- Példa
  - r = n/5
  - 20% hiba ½ valószínűséggel

[Charikar, Chaudhuri, Motwani, Narasayya 2000]

Adatfolyam determinisztikus alsó korlát

Tétel: Determinisztikus adatfolyam algoritmus memóriaigénye $\Omega(n \log m)$

Bizonyítás:

- Tételezzük fel – determinisztikus A $o(n \log m)$ bitet használ
- Válasszunk – inputot, U, mérete $n<m$
- S –A állapota az input elolvasása után
- Ellenőrizhetjük, hogy bármelyik $x_i \in U$ úgy, hogy A megkapja x_i-t következő inputnak
  - D(X) nem nő pontosan akkor, ha $x_i \in X$
- Iinformáció-elmélet – U visszaállítható S-ből
- Tehát – $\binom{m}{n}$ állapot, $\Omega(n \log m)$ bit

Véletlen közelítés

- Alsó korlát megenged randomizált vagy közelítő algoritmusokat
- SM Algoritmus – Fix t-re D(X) >> t?
  - hash függvény *h: U →[1..t]*
  - Kezdő válasz NEM
  - Minden x_i -re, ha *h*(x_i) = *t,* akkor a válasz IGEN
- Tétel:
  - Ha D(X) < t, P[SM kimenete NEM] > 0.25
  - Ha D(X) > 2t, P[SM kimenete NEM] < 0.136 = 1/e^2
- Figyelem – 1 bit memória kell csak!

[Indyk-Motwani 1998]

Hiba csökkentése

- 1 bittel →
  valószínűleg elkülöníthető D(X) < t és D(X) > 2t
- O(log 1/δ) független hash függvény → hibavalószínűség tetszőlegesen kicsi δ>0 lehet
- O(log n) független hash függvény t = 1, 2, 4, 8 …, n esetén → D(X) becsülhető 2 szorzón belül
- A 2 itt tetszőleges konstans → (1+ε) esetén hiba ε
- Ellenőrizni – D(X) (1±ε) szorzón belül (1-δ) valószínűséggel becsülhető

$$O\left(\log \frac{n}{\varepsilon^2} \times \log \frac{1}{\delta}\right)$$

tárban.

Leszámlálás vagy mintavételezés

- Nehezebb feladat
  - Adatok sok értékkel – X csak egy attribútum
  - Adatbázis-feladatok – select, join, …
  - Elosztott algoritmusok – adatfolyamok kombinációja
- Előző algoritmus
  - Kicsi hiba
  - De csak számol – egyik fenti feladatot sem tudja megoldani
- Mintavételezés
  - Megtarthatja a többi attribútumot is – fenti feladatok kezelhetők
  - De előbb láttunk egy nagyon rossz alsó korlátot ☹

A Distinct Sampling módszer

- A két világból a legjobbat akarjuk
  - Nagy pontosság
  - "distinct sample" kiválasztása az adatfolyamból
  - De minden elemet elolvasunk!
- Ötlet
  - Hash $\rightarrow$ véletlen "prioritás" az értékekre
  - Proiritás a kezdő 0-k száma h(x) bináris felírásában
  - Az $O\left(\varepsilon^{-2}\log\delta^{-1}\right)$ legnagyobb prioritású eddig látott érték megtartása
  - Minta teljes adattartalommal
  - ε relatív hiba $1-\delta$ valószínűséggel

[Gibbons 2001]

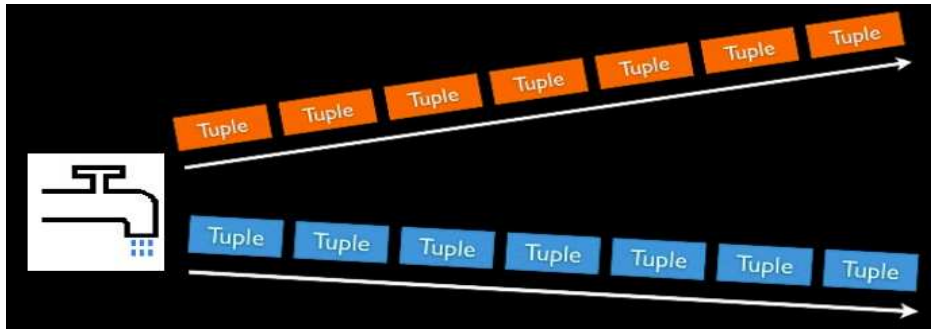
A Distinct Sample algoritmus

- Paraméter – memória-méret $\mathrm{M} = O\left(\varepsilon^{-2} \log \delta^{-1}\right)$
- Indítás – *cur_lev*←0; S←üres
- Minden x inputra
  - L ← *h(x)*
  - Ha L>*cur_lev* akkor *x* hozzáadása S-hez
  - Ha |S| > M
    - S-ből minden pontosan *cur_lev* szintű elem törlése
    - *cur_lev* ← *cur_lev* +1
- Eredmény $2^{cur\_lev} \times |S|$

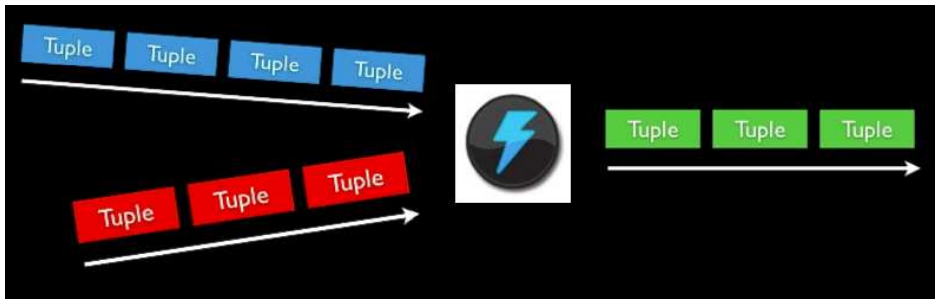
Adatfolyam rendszer: Twitter Storm

1. Adatfolyam
   - Végtelen „tuple” sorozat

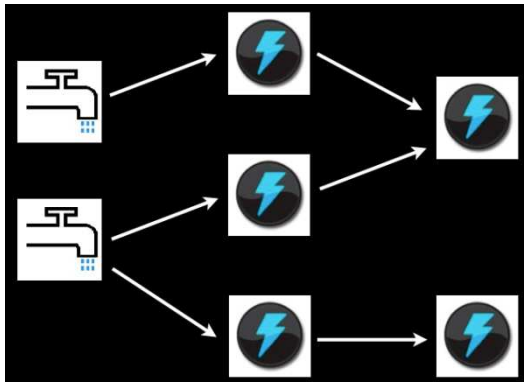
2. Spout
   - Forrás
   - Pl. Twitter streaming API



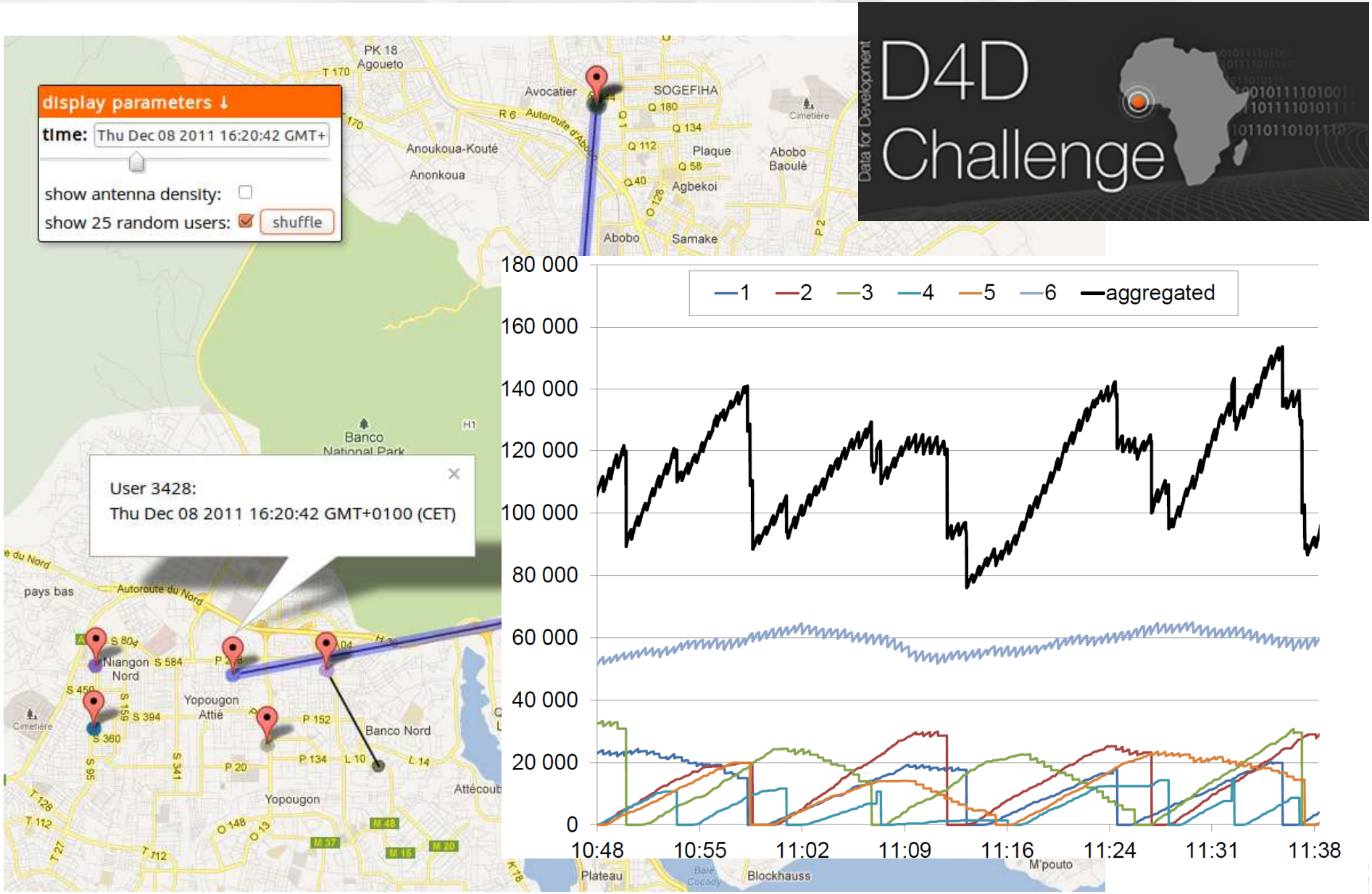
3. Bolt
   - Input folyamot olvassa, feldolgozza, új folyamot ír
   - Pl. függvény, szűrő, aggregátor, join



4. Topológia
   - Spout és bolt DAG (irányított körmentes gráf)



Valós idejű mobilitás előrejelzés



További információ

- NoSQL bevezető – *www.intertech.com/resource/usergroup/NoSQL.ppt*
- Key-value stores
  - BerkeleyBD – nem osztott
  - Voldemort – *behemoth.strlen.net/~alex/voldemort-nosql_live.ppt*
  - Cassandra, Dynamo, …
  - Hadoop alapon is létezik (lent): HBase
- Hadoop – *www.cca08.org/files/slides/Owen-OMalley-Hadoop – CCA 2008.ppt*
- HBase – *datasearch.ruc.edu.cn/course/cloudcomputing20102/slides/Lec07.ppt*
- Mahout – *cwiki.apache.org/MAHOUT/faq.data/Mahout Overview.ppt*
- Miért kell más? Mi más kell?
  - Bulk Synchronous Parallel
  - Graphlab – *http://graphlab.org/home/publications/*
  - MOA – *http://www.slideshare.net/abifet/moa-5636332/download*
- Streaming
  - Storm - *http://engineering.twitter.com/2011/08/storm-is-coming-more-details-and-plans.html*
  - S4 – *http://www.slideshare.net/alekbr/s4-stream-computing-platform*
- Saját előadás-sorozatunk:
  https://dms.sztaki.hu/hu/letoltes/elosztott-technologiak-eloadassorozat

Kérdések?

Benczúr András
Informatika Kutatólabor

“Big Data” lab

http://datamining.sztaki.hu/

benczur@sztaki.mta.hu